Flex static code analysis

Unique rules to find Bugs, Security Hotspots, and Code Smells in your FLEX code

All rules 76 | Vulnerability 5 | Bug 9 | Security Hotspot 1 | Code Smell 61

Tags | Search by name...

Security.allowDomain(...) should only be used in a tightly focused manner
Vulnerability

The flash.system.Security.exactSettings property should never be set to false
Vulnerability

Dynamic classes should not be used
Code Smell

"LocalConnection" should be configured to narrowly specify the domains with which local connections to other Flex application are allowed
Vulnerability

"default" clauses should be first or last
Code Smell

Event types should be defined in metadata tags
Code Smell

Event names should not be hardcoded in event listeners
Code Smell

The special "star" type should not be used
Code Smell

Variables of the "Object" type should not be used
Code Smell

Methods should not be empty
Code Smell

Constant names should comply with a naming convention
Code Smell

All branches in a conditional structure should not have exactly the same implementation
Bug

Classes that extend "Event" should

"for" loop stop conditions should be invariant

Analyze your code

Code Smell   Major   pitfall

A `for` loop stop condition should test the loop counter against an invariant value (i.e. one that is true at both the beginning and ending of every loop iteration). Ideally, this means that the stop condition is set to a local variable just before the loop begins.

Stop conditions that are not invariant are slightly less efficient, as well as being difficult to understand and maintain, and likely lead to the introduction of errors in the future.

This rule tracks three types of non-invariant stop conditions:

- When the loop counters are updated in the body of the `for` loop
- When the stop condition depend upon a method call
- When the stop condition depends on an object property, since such properties could change during the execution of the loop.

Noncompliant Code Example

```
for (var i = 0; i < 10; i++) {
  ...
  i = i - 1; // Noncompliant
  ...
}

for (var i = 0; i < getMaximumNumber(); i++) {...}
```

Compliant Solution

```
int stopCondition = getMaximumNumber();
for (var i = 0; i < stopCondition; i++) {...}
```

Available In:
sonarcloud sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
Privacy Policy

override "Event.clone()"
Bug

Constructors should not dispatch events

Bug

"ManagedEvents" tags should have companion "Event" tags

Bug

Objects should not be instantiated inside a loop

Code Smell

Two branches in a conditional structure should not have exactly the same implementation

Code Smell

Constructor bodies should be as lightweight as possible

Code Smell

Only "while", "do" and "for" statements should be labelled

Code Smell

Statements, operators and keywords specific to ActionScript 2 should not be used

Code Smell

"for" loop stop conditions should be invariant

Code Smell

Unused function parameters should be removed

Code Smell